CS60045 Artificial Intelligence Autumn 2023

Propositional Logic

Logical Reasoning

- So far, our intelligent agents can handle a variety of searches. That alone is not sufficient.
- AI agents must be capable of taking decisions and planning for the future.
- AI agents should often be able to do these new tasks in presence of uncertainty and/or imprecise information.
- The actions are guided by a **knowledge base** which are available from the beginning and/or gathered during the process.
- Whatever it does, it should be **logical**.
- Various types of logic
  - Propositional logic
  - Predicate logic
  - Temporal logic
  - Fuzzy logic

The language of propositional logic

Propositions

A proposition is a declarative statement that is either *true* or *false*, and nothing else.

Examples

It is raining.
2 is an even number.
Either I win or you lose.
If $2 + 2 = 5$, then $3 + 3 = 6$.

Non-examples

It may rain.
Is it raining?
x is an even number.
Please try to win.

The language of propositional logic

Atomic propositions

These are propositions that cannot be decomposed. These are also called **atoms**.

Composite propositions

These are recursively constructed as follows.

- **[Negation]** If P is a proposition, then $\neg P$ is a proposition.
- **[Disjunction]** If P and Q are propositions, then $P \vee Q$ is also a proposition.
- **[Conjunction]** If P and Q are propositions, then $P \wedge Q$ is also a proposition.
- **[Implication]** If P and Q are propositions, then $P \Rightarrow Q$ is also a proposition.
- **[Biconditional]** If P and Q are propositions, then $P \Leftrightarrow Q$ is also a proposition.
- **[Disambiguation]** If P is a proposition, then (P) is also a proposition.

Propositions are also called (propositional) **sentences** and **well-formed formulas** (**wffs**).

A **literal** is either an atomic proposition (a **positive literal**) or its negation (a **negative literal**).

Truth tables

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

Two wffs are called **equivalent** if they have the same truth table.

Examples

- **[Implication elimination]** $P \Rightarrow Q \equiv \neg P \vee Q$.
- **[Contraposition]** $P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P$.
- **[Biconditional elimination]** $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (P \Leftarrow Q)$. ($P \Leftarrow Q$ means $Q \Rightarrow P$.)
- **[De Morgan's laws]** $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$ and $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$.
- **[Distributive laws]** $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$ and $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$.

Example: Reach the center of the labyrinth

While walking in a labyrinth, you find yourself in front of three possible roads. The road on your left is paved with gold, the road in front of you is paved with marble, and the road on your right is made of small stones. Each road is protected by a guard. You talk to the guards, and this is what they tell.

- The guard of the gold road: “This road will bring you straight to the center. Moreover, if the stones take you to the center, then also the marble takes you to the center.”
- The guard of the marble road: “Neither the gold nor the stones will take you to the center.”
- The guard of the stone road: “Follow the gold, and you will reach the center. Follow the marble, and you will be lost.”

Given that all the guards are liars, which road should you take to reach the center?

Reach the center of the labyrinth: Formulation

Atomic propositions

G	The gold road leads to the center
M	The marble road leads to the center
S	The stone road leads to the center

Composite propositions made from the statements of the guards

GG	$G \wedge (S \Rightarrow M)$
GM	$\neg G \wedge \neg S$
GS	$G \wedge \neg M$

Knowledge base

$KB \quad \neg GG \wedge \neg GM \wedge \neg GS$

Which of the following is/are true?

- $KB \Rightarrow G$
- $KB \Rightarrow M$
- $KB \Rightarrow S$

Reach the center of the labyrinth: Solution using truth tables

F and T are often written as 0 and 1.

G	M	S	$\neg GG$	$\neg GM$	$\neg GS$	KB
0	0	0	1	0	1	0
0	0	1	1	1	1	1
0	1	0	1	0	1	0
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	1	1	1	0	0
1	1	0	0	1	1	0
1	1	1	0	1	1	0

Conclusions

- The gold road will certainly not take you to the center.
- The marble road may or may not lead you to the center.
- The stone road will definitely lead you to the center.

Truth assignments

A **truth assignment** is giving a T or F value to each atomic proposition.

A truth assignment is also called a **model** or an **interpretation**.

For every model, the truth values of composite propositions built from the atomic propositions can be obtained.

A wff is called **satisfiable** if it evaluates to T for at least one truth assignment.

A wff is called **valid** or a **tautology** if it evaluates to T for all truth assignments.

A wff is called a **contradiction** if it evaluates to F for all truth assignments.

Entailment: P entails Q if whenever P is true, Q is also true. P entails Q if and only if $P \Rightarrow Q$ is a tautology.

Contradiction: $P \Rightarrow Q \equiv \neg P \vee Q$. The negation of this is $\neg(P \Rightarrow Q) \equiv P \wedge \neg Q$. Therefore P entails Q if and only if $P \wedge \neg Q$ is a contradiction.

Propositional theorem proving

Input: A knowledge base consisting of propositions $P_1, P_2, P_3, \ldots, P_n$, and a goal proposition Q.

Output: T or F depending on whether $P_1 \wedge P_2 \wedge P_3 \wedge \cdots \wedge P_n$ entails Q or not.

Let M be a proof mechanism (procedure).

M is called **sound** if whenever M outputs T, $P_1 \wedge P_2 \wedge P_3 \wedge \cdots \wedge P_n$ entails Q.

M is called **complete** if whenever $P_1 \wedge P_2 \wedge P_3 \wedge \cdots \wedge P_n$ entails Q, M outputs T.

M is called **total** if it gives some output on all instances.

Notes

- Soundness guarantees that if M outputs T, then the entailment is true.
- Completeness guarantees that if the entailment is true, M will eventually output T.
- Neither soundness or completeness guarantees that M is bound to give the answer F when the entailment is not true. In this case, M may actually fail to give any answer.
- M is an **algorithm** if it is sound and complete and total.

Propositional theorem proving is decidable

A computational problem is called **decidable** if it has an algorithm.

A computational problem is called **semidecidable** if it has a sound and complete procedure.

We can decide whether $P_1 \wedge P_2 \wedge P_3 \wedge \cdots \wedge P_n$ entails Q by constructing the truth table of $P_1 \wedge P_2 \wedge P_3 \wedge \cdots \wedge P_n \Rightarrow Q$.

If there are n atomic propositions involved, then the truth table has 2^n rows.

So the truth-table method is not an *efficient* algorithm for propositional theorem proving.

Rules of inference

Proceed by making a sequence of small logical steps.

Use the laws of propositional calculus (commutativity, associativity, distributivity, De Morgan's laws).

The following rules are also often handy.

- **Modus Ponens:** $P \Rightarrow Q$ and P entail Q.
- **Modus Tollens:** $P \Rightarrow Q$ and $\neg Q$ entail $\neg P$.
- **And elimination:** $P \wedge Q$ entails P.
- **Or introduction:** P entails $P \vee Q$.
- **And introduction:** P and Q entail $P \wedge Q$.
- **Syllogism:** $P \Rightarrow Q$ and $Q \Rightarrow R$ entail $P \Rightarrow R$.
- **Disjunctive syllogism:** $P \vee Q$ and $\neg P$ entail Q.

Reach the center of the labyrinth: Solution by the rules of inference

The statements of the guards entail S

Using the rules of propositional calculus, we rewrite $\neg GG$ as

$$\begin{aligned} \neg GG &\equiv \neg\Big(G \wedge (S \Rightarrow M)\Big) \equiv \neg G \vee \neg(\neg S \vee M) \equiv \neg G \vee (S \wedge \neg M) \\ &\equiv (\neg G \vee S) \wedge (\neg G \vee \neg M). \\ \neg GM &\equiv \neg(\neg G \wedge \neg S) \equiv G \vee S. \end{aligned}$$

And elimination from $\neg GG$ gives $\neg G \vee S$. By And introduction, we get

$$(\neg G \vee S) \wedge (G \vee S).$$

By distributivity, we then have

$$(\neg G \wedge G) \vee S \equiv \mathsf{F} \vee S \equiv S.$$

Reach the center of the labyrinth: More about the rules of inference

The statements of the guards entail $\neg G$

And elimination from $\neg GG$ gives $\neg G \vee \neg M$. Moreover, $\neg GS$ simplifies to $\neg G \vee M$. And introduction gives

$$(\neg G \vee \neg M) \wedge (\neg G \vee M) \quad \equiv \quad \neg G \vee (\neg M \wedge M) \quad \equiv \quad \neg G \vee \mathsf{F} \quad \equiv \quad \neg G.$$

The statements of the guards entail neither M nor $\neg M$

- How do we *prove* this?
- We may try for many steps to discover that we cannot derive M or $\neg M$.
- We will report failure when we are sufficiently tired.
- But that is not any proof.

Conclusion: The rules of inference do not immediately give an *algorithm*.

The conjunctive normal form (CNF)

- A **literal** is either an atomic proposition or a complement of an atomic proposition.
- A **clause** is a disjunction (or) of one or more literals.
- A wff is in **CNF** is it a conjunction (and) of clauses.
- **CNFSAT** is the problem of deciding whether a wff in the CNF is satisfiable (that is, T for at least one truth assignment of the atomic propositions).
- **Theorem:** CNFSAT is NP-complete.
- **SAT** is the (more general) problem of deciding whether a wff is satisfiable.
- **Theorem:** SAT is NP-complete.
- **TAUTOLOGY** (resp. **CONTRADICTION**) is the problem of deciding whether a wff is a tautology (resp. a contradiction).
- **Theorem:** TAUTOLOGY is coNP-complete.
- **Theorem:** CONTRADICTION is coNP-complete.

Every wff can be converted to CNF

We prove this by induction on the length of the wff ϕ.

Base: Every atom is in the CNF.

Induction:

Let $\phi = \alpha \wedge \beta$. By induction, α and β can be converted to CNF.

Let $\phi = \alpha \vee \beta$. By induction, we can write $\alpha = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ and $\beta = D_1 \wedge D_2 \wedge \cdots \wedge D_n$, where each C_i and each D_j are clauses. By distributivity, we can write ϕ as $\bigwedge_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant n}} (C_i \vee D_j)$.

Let $\phi = \neg\alpha$. By induction, we can write $\alpha = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each C_i is a clause. By De-Morgan's law, $\neg\alpha = (\neg C_1) \vee (\neg C_2) \vee \cdots \vee (\neg C_m)$. Now, use the case $\phi = \alpha \vee \beta$. ◀

Note: We can write the truth table of $\neg\phi$, get a DNF expression for $\neg\phi$ from the truth table, and then complement it. But we prefer to avoid truth tables for there exponential sizes.

Example: $\neg(p \Rightarrow q) \vee (q \Rightarrow \neg r) \equiv \neg(\neg p \vee q) \vee (\neg q \vee \neg r) \equiv (p \wedge \neg q) \vee (\neg q \vee \neg r) \equiv (p \vee \neg q \vee \neg r) \wedge (\neg q \vee \neg q \vee \neg r) \equiv (p \vee \neg q \vee \neg r) \wedge (\neg q \vee \neg r)$.

Resolution of clauses: A new rule of inference for CNF

Let C and D be two clauses containing a pair of complementary literals. Let C contain the literal ℓ, and D the literal $\neg\ell$. Then, we can write $C = \ell \vee C'$ and $D = \neg\ell \vee D'$. If C' or D' is empty, take it as F. In this case, the **resolvent** of C and D is defined to be

$$\mathrm{Res}_\ell(C, D) = C' \vee D'.$$

We say that $\mathrm{Res}_\ell(C, D)$ is obtained by **resolving** C and D upon the literal ℓ (or $\neg\ell$).

Examples

- $\mathrm{Res}((p \vee q \vee \neg r), (p \vee \neg q \vee s)) = p \vee \neg r \vee s$.
- $\mathrm{Res}((p \vee \neg q \vee r), (\neg p \vee q \vee s \vee \neg t)) = p \vee r \vee \neg p \vee s \vee \neg t \equiv \mathsf{T}$.
- $\mathrm{Res}(\ell, \neg\ell \vee p \vee \neg q) = p \vee \neg q$.
- $\mathrm{Res}(\ell, \neg\ell) = \mathsf{F}$.

Properties of resolution

Theorem: C and D entail $\mathrm{Res}(C, D)$.

Proof We have $C \equiv \neg C' \Rightarrow \ell$ and $D \equiv \ell \Rightarrow D'$. By syllogism, we have $\neg C' \Rightarrow D' \equiv C' \vee D'$. ◀

Resolution is sound. This theorem says that it never makes an error in the entailment procedure.

Resolution is not complete. Let p, q be atomic propositions. Then, p and q entail $p \vee q$. However, we cannot resolve p and q, because there is no literal to resolve upon.

Despite this badness, resolution has nice properties. Take a wff ϕ in CNF. Let S be the set of clauses in ϕ. Keep on resolving pairs C, D of clauses in S, and adding $\mathrm{Res}(C, D)$ to S. Stop when the resolutions of clauses in S cannot give any new clause. The final value of S is called the **resolution closure** of ϕ (or of the set of clauses in ϕ). We denote this by $RC(\phi)$.

By an abuse of notations, we call this process resolution too.

The process stops after finitely many steps (because ϕ contains only finitely many literals, and each clause in S is composed of a subset of these literals).

Ground resolution theorem

Theorem: ϕ is a contradiction if and only if $RC(\phi)$ contains the empty clause (that is, F).

Proof [If] ϕ entails all the clauses in $RC(\phi)$, so the presence of the empty cause in $RC(\phi)$ implies $\phi \Rightarrow$ F, that is, ϕ is a contradiction.

[Only if] Suppose that $RC(\phi)$ contains only non-empty clauses. Let the atoms in ϕ be $p_1, p_2, \ldots, p_n$. Define S_i to be those clauses in $RC(\phi)$, in which p_i or $\neg p_i$ is the largest-numbered literal. We take $S_0 = \emptyset$. We have $RC(\phi) = \bigcup_{i=1}^{n} S_i$. In the sequence $i = 1, 2, \ldots, n$, we inductively keep on supplying a truth value to p_i so as to make all the clauses in S_i true.

For each clause in S_i, if the truth assignments of $p_1, p_2, \ldots, p_{i-1}$ already satisfy the clause, remove the clause from S_i. If S_i becomes empty, assign any truth value to p_i. Otherwise denote by S_i^+ (resp. S_i^-) the (remaining) clauses in S_i, that contain p_i (resp. $\neg p_i$). If both S_i^+ and S_i^- are non-empty, then pick C from S_i^+ and D from S_i^-. Since $RC(\phi)$ is closed under resolution, $\text{Res}(C, D) \in S_j$ for some $j < i$ ($j \neq 0$ because $S_0 = \emptyset$). By the choices of C and D, $\text{Res}(C, D)$ evaluates to F, a contradiction to the induction hypothesis. Therefore either S_i^+ or S_i^- must be empty. If S_i^- is empty, take $p_i =$ T. If S_i^+ is empty, take $p_i =$ F. ◀

Propositional theorem proving by resolution

The basic task is to prove whether the knowledge base consisting of propositions $P_1, P_2, \ldots, P_n$ entails the goal Q.

This is logically equivalent to proving that $(P_1 \wedge P_2 \wedge \cdots \wedge P_n) \wedge \neg Q$ is a contradiction.

This is akin to proof by contradiction. Assume that all the propositions in the knowledge base to be true. Assume also that the goal is false. Deduce that these assumptions lead to F.

This ***algorithm*** is called **resolution refutation**. Refutation is an AI nickname for contradiction.

All you need to do is:

- Convert $(P_1 \wedge P_2 \wedge \cdots \wedge P_n) \wedge \neg Q$ to CNF. Call this wff ϕ.
- Apply the resolution procedure on ϕ until either the empty clause is obtained as a resolvent (return T in this case) or there are no further resolution possibilities (return F in this case).

Resolution refutation for finding the center of the labyrinth

[$\neg GG$, $\neg GM$, and $\neg GS$ entail S]

$\neg GG \equiv (\neg G \vee S) \wedge (\neg G \vee \neg M)$ gives the clauses $\boxed{C_1 = \neg G \vee S}$ and $\boxed{C_2 = \neg G \vee \neg M}$. $\neg GM$ gives the clause $\boxed{C_3 = G \vee S}$. $\neg GS$ gives the clause $\boxed{C_4 = \neg G \vee M}$. Also take the clause $\boxed{C_5 = \neg S}$.

Resolve C_1 and C_3 on G to get $\boxed{C_6 = S \vee S = S}$. Resolve C_5 and C_6, to get the empty clause.

[$\neg GG$, $\neg GM$, and $\neg GS$ entail $\neg G$]

Start with the clauses C_1, C_2, C_3, C_4 as above, but with $\boxed{C_5 = \neg(\neg G) = G}$.

Resolve C_1 and C_3 on G to get $\boxed{C_6 = \neg G \vee \neg G = \neg G}$. Resolve C_5 and C_6, to get the empty clause.

Resolution refutation for finding the center of the labyrinth

[$\neg GG$, $\neg GM$, and $\neg GS$ does not entail M]

Start with the clauses $C_1 = \neg G \vee S$, $C_2 = \neg G \vee \neg M$, $C_3 = G \vee S$, $C_4 = \neg G \vee M$, and $C_5 = \neg M$.

First level of resolvents

$C_6 = \text{Res}_G(C_1, C_3) = S$, $C_7 = \text{Res}_G(C_2, C_3) = \neg M \vee S$, $C_8 = \text{Res}_G(C_3, C_4) = M \vee S$
$C_9 = \text{Res}_M = \neg G$ ($\text{Res}_M(C_4, C_5)$ is the same as C_9)

Second level of resolvents

No new resolvent can be added (for example, $\text{Res}_G(C_9, C_3) = S = C_6$, $\text{Res}_M(C_7, C_4) = C_1$, $\text{Res}_M(C_7, C_8) = C_6$).

Truth assignment: Let us assign G, M, S in that order.

$S_1 = \{C_9\}$. So take $G = \text{F}$.
$S_2 = \{C_2, C_4, C_5\}$. C_2 and C_4 are already satisfied by $G = \text{F}$. To satisfy C_5, take $M = \text{F}$.
$S_3 = \{C_1, C_3, C_6, C_7, C_8\}$. C_1 and C_7 are already satisfied. C_3, C_6, C_8 are all satisfied by $S = \text{T}$.

Ordering strategies

Zeroth-level resolvents: Original clauses in ϕ.
First-level resolvents: Resolvents of zeroth-level resolvents.
Second-level resolvents: Resolvents of first-level resolvents with zeroth- and first-level resolvents.
i-th-level resolvents: Resolvents of $(i-1)$-th level resolvents and j-th level resolvent for $j \leqslant i-1$.

Breadth-first strategy

Start with the zeroth-level resolvents, then generate all first-level resolvents, then all second-level resolvents, and so on.

Depth-first strategy

Recursively generate a first-level resolvent, then a second-level resolvent, then a third-level resolvent, and so on.

Unit-preference strategy

A clause with a single literal is called a **unit clause**. Prefer unit clauses while computing resolvents whenever applicable.